REMARKS

Claims 1 to 6, 8 to 13, 15 to 19, 24, 26 to 31, 33 to 38, and 40 to 42 were pending in the application at the time of the final examination. Claims 1, 12, 13, 15, 24, 26, 37, 38, and 40 to were objected to for informalities. Claims 1 to 6, 8 to 13, 15 to 19, 24, 26 to 31, 33 to 38, and 40 to 42 stand rejected as obvious.

Claims 1, 12, 13, 24, 26, 37, 38, and 40 have been amended to correct the informality noted by the Examiner. Applicants respectfully request reconsideration and withdrawal of the objection to each of Claims 1, 12, 13, 24, 26, 37, 38, and 40.

Applicants respectfully request entry of the amendments to the claims. The amendments correct a typographical error and so entry does not require consideration of new issues or a new search. In view of the following remarks, Applicants respectfully submit that entry is proper under Rule 116 because the amendments place the application in condition for allowance. If the Examiner should disagree, entry is requested to narrow the issues for appeal.

Claims 1 to 6, 8 to 13, 15 to 19, 24, 26 to 31, 33 to 38, and 40 to 42 stand rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6,829,770, hereinafter referred to as Hinson, in view of U.S. Patent No. 6,694,506, hereinafter referred to as LeBlanc and Admitted Prior Arts(APA).

Applicants respectfully traverse the obviousness rejections of Claims 1, 26 and 40. To make a prima facie obviousness rejection, the MPEP directs:

> **BASIC CONSIDERATIONS WHICH APPLY TO OBVIOUSNESS REJECTIONS**
> When applying 35 U.S.C. 103, the following tenets of patent law mmust be adhered to:
> (A) The claimed invention must be considered as a whole;
> (B) The references must be considered as a whole and must suggest the desirability and thus the obviousness of making the combination;

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

> (C) The references must be viewed without the benefit of impermissible hindsight vision afforded by the claimed invention; and
> (D) Reasonable expectation of success is the standard with which obviousness is determined.

MPEP § 2141, 8th Ed., Rev. 3, p. 2100-125 (August 2005). It is noted that this directive stated "the following tenets . . . **must be adhered to**." Accordingly, the failure of the Examiner to adhere to any one of these tenets means that a prima facie obviousness rejection has not been made.

The final rejection failed to adhere to multiple of these tenets and so a prima facie obviousness rejection has not been made. As demonstrated more completely below, the claimed invention has not been considered as a whole; the references have not been considered as a whole; and the references do not suggest the desirability of making the combination. Pieces of the references have been extracted and selectively interpreted in view of Appellants' claims. Finally, there was no explanation of how the primary reference would work for its intended purpose following the modification.

In the rejection of Claim 1, the Examiner first stated in part:

> . . . an event (provides an event class object to distribute information produced by a publisher to one or more subscribers, lines 41-43 column 11) in a object facility repository(142, Fig. 7)
>
> the event having an event type (to subscribe to a particular outgoing event interface method . . ., lines 40-53, column 13; the type of object to be retrieved from the event objects store 140, lines 60-61 column 16 . . .
>
> creating an event object for the event(event object 102, Fig. 5; event objects within the storage 142, Fig. 7), the event object corresponding to the event sub-type (class of the event objects having methods, line 8-41 column 12);

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

Final Office Action dated 07/12/2006, pages 2 and 3.

Applicants respectfully note that these Claims recite "an event" and "the event," so that one event is being recited in the claim. Nevertheless, the rejection considers the claim piecemeal and cites to multiple different parts of Hinson with respect to this event.

First the rejection cites to Col. 16, lines 60 and 61 as defining the event type. However, when Hinson is considered as a whole, i.e., Col. 16, lines 60 and 61 are taken in context, Hinson taught:

> FIGS. 21-31 are program listings 230-241 of various interfaces defined per the COM+ Events system. These system-defined interfaces are exposed by various system-provided objects, as well as by the publisher 102 and subscriber 106 objects written by the application developer or other user of the illustrated event communications model 100 (FIG. 5) for integration into the model.
>
> The program listing 230 of FIG. 21 defines an IEventSystem interface that is exposed by the COM+ Events system 140 (FIG. 7). The IEventSystem interface includes the methods, "Query( )," "Store( )," "Remove( )," and "EventObjectChangeEventClassID( )." The "Store( )" method is used to install objects into the event objects store 142 (FIG. 7), such as the publisher 104 and the event class 102 supporting the publisher's outgoing-event interface (as demonstrated in the program listing 164 of FIG. 13 of the Stock Exchange application example), as well as the subscriptions 120 (as demonstrated in the program listing 192 of FIG. 19).
>
> The "Query( )" method is used to retrieve objects from the event objects store 140. The method takes two input parameters ("ProgID" and "wszQueryCriteria") and two output parameters ("errorIndex" and "ppInterface"). **The "ProgID" parameter identifies the type of object to be retrieved from the event objects store 140.** The valid type values that can be specified in this parameter are listed in the header file statements shown in the program listing 242 of FIG. 32. (Emphasis Added. Bold Emphasis is portion cited in rejection).

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

Thus, the rejection confuses "an event type" with "objects." Further, Hinson expressly stated that the valid values of "ProgID" are given in Fig. 32. Fig. 32 defines

. . . PROGID_EventPublisher . . .
. . . PROGID_EventClass . . .
. . . PROGID_EventSubcription
. . .
. . . PROGID_EventPublisherCollection
. . .
. . . PROGID_EventClassCollection
. . .
. . . PROGID_EventSubscriptionCollection

These are objects in the COM Event System 140 of Hinson. This fails to provide any teaching or suggestion of "the event having an event type and an event sub-type," as recited in these claims.

The rejection, in the next recitation of event type and event sub-type in these claims, reduced the claim language to a gist as quoted above, "creating an event object for the event." The claims do not recite creating an event object just for some event but rather a specific event, "the event object corresponding to the event sub-type." The event types initially cited in the rejection do not have subtypes and so the rejection switches focus to event object 102, i.e., the definition changes based on the location in the claim.

In this part of the rejection, the event sub-type is reduced to "class of the event objects having methods, line 8-41 column 12." Again, at best, a general gist is being rejected. The event object is described by Hinson as "The event class object 102 thus is a system-supplied object." The rejection has failed to cite any teaching or suggestion in the cited portion of Hinson of a sub-type of an event type for the

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

event. In particular, there is no citation to a suggestion or teaching in Hinson of sub-typing a particular event type based on the methods included in the event. The only basis for this is Applicants' claims language.

The rejection further stated in part:

> Hinson does not explicitly teach the object facility repository is a meta object facility repository. However, Hinson teaches (lines 50-57 column 7) that the invention can be implemented in combination with other program modules that implement particular abstract data types. Therefore one of ordinary skill in the art could conclude that the particular abstract data type could be metadata defining the structure of data objects and the object facility repository of Hinson could be a meta object facility repository.

This statement mischaracterizes the reference, mischaracterizes abstract data type, mischaracterized metadata, and fails to consider Hinson as a whole yet again. There is no citation to any reference with respect to the level of skill in the art interpreting abstract data type or metadata. Therefore, the conclusory statement concerning what one of skill in the art could conclude after reading "abstract data types" is not supported by any prior art teaching.

An abstract data type is not interpreted by those of skill in the art as a type of data as inferred in this part of the rejection. Hinson stated:

> Generally, **program modules include routines, programs, components, data structures, etc.** that perform particular tasks or **implement particular abstract data types**

Thus, Hinson taught program modules include elements that implement abstract data types. An implementation of an abstract data type is needed because an abstract data type is "A set of data values and associated operations that are

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

precisely specified independent of any particular implementation." Paul E. Black, "abstract data type", in Dictionary of Algorithms and Structures [online], Paul E. Black, ed., U.S. National Institute of Standards and Technology, 10 February 2005. Available from: http:/www.nist.gov/dads/HTML/abstactDataType.html. A copy of this document is included as Attachment 1. Applicants note that the date of this document is after the filing date of the above application. However, the definition is consistent with that used by those of skill in the art and the use of the term in Hinson. Accordingly, to the extent that the rejection infers that Hinson is describing a type of data, the rejection is inconsistent with the level of skill in the art.

Further, the specification defines "Metadata is information about data, or simply data about data." Accordingly, when the level of skill in the art is considered it is error to equate metadata with abstract data type as was done in the rejection and further, there is no basis other than Examiner argument to support the conclusion as noted above.

Next, Hinson in context taught:

> Exemplary Operating Environment
>
> FIG. 1 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. While the invention will be described in the general context of computer-executable instructions of a computer program that runs on a computer, those skilled in the art will recognize that the invention also may be implemented in combination with other program modules. Generally, program modules include routines, programs, components, data structures, etc. that perform particular tasks or implement particular abstract data types.

Thus, the "other modules" are those in an operating system. Hinson makes this clear by stating:

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

> With reference now to FIG. 4, the computer 20 (FIG. 1) executes component applications that are developed as a package of component application objects. In the illustrated embodiment of the invention, the component application objects conform to the Microsoft Component Object Model ("COM") specification (i.e., are implemented as a "COM Object" 60) and executed using the COM+ services of the Microsoft Windows NT Server 5.0 operating system as stated above, but alternatively may be implemented according to other object standards (including the CORBA (Common Object Request Broker Architecture) specification of the Object Management Group) and executed under object services of another operating system. The COM specification defines binary standards for objects and their interfaces which facilitate the integration of software components into applications. (For a detailed discussion of COM and OLE, see Kraig Brockschmidt, Inside OLE, Second Edition, Microsoft Press, Redmond, Wash. (1995)).

Thus, when considered as a whole, Hinson expressly taught that either COM or CORBA was used with an operating system to implement the invention. Accordingly, the statement concerning "other modules" by Hinson provides no basis for replacing the system of Hinson with a completely different system. Further, the events that are fired by the publisher in Hinson are based on something associated with the publisher. The rejection has failed to cite any teaching or suggestion of firing events based on anything that happens in object store 142 of Hinson. Accordingly, the modification changes the principles of operation of Hinson. The MPEP indicates that such a change means that a prima facie obviousness rejection has not been made.

The information relied upon in LeBlanc fails to overcome the shortcomings noted above. Accordingly, the combination of references fails to suggest or disclose the method of Claim 1. Applicants request reconsideration and withdrawal of the obviousness rejection of each of Claims 1, 26, and 40.

Claims 2 to 6 and 8 to 11 depend from Claim 1 and so distinguish over the combination for at least the same reasons

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

as given above for Claim 1. Further, for example, the "bitmask" of Claim 6 has been reduced to a gist. "Bitmask" or "bit mask" does not appear in Hinson and so Hinson fails to suggest or describe the combination event source interface of Claim 6. Applicants respectfully request reconsideration and withdrawal of the obviousness rejection of each of Claims 2 to 6 and 8 to 11.

With respect to the obviousness rejection of Claim 12, the Examiner relies upon the rejection of Claims 1 and 6. However, Claim 12 recites "wherein said event type is included in a plurality of event types including an instance event, a class event, an association event . . . " Neither Claim 1 nor Claim 6 includes this plurality and so the rejection stated:

> Hinson further teaches event types including an instance event (line 58 column 1 to line 10 column 2 . . . an associate event and a combination of events (line 58 column 1 to line 10 column 2).

Applicants respectfully note "event type" as interpreted in this part of the rejection is different from the interpretation of event type in the rejection of Claim 1, which is used to form the basis of the rejection of Claim 6. See also the rejection of Claim 5 that utilizes but yet another different definition. This demonstrates yet further that the definition selected changes depending on the part of the claim and the claim considered.

Further, Col. 1, line 58 to Col. 2 line 10 of Hinson stated:

> In accordance with object-oriented programming principles, the component application is a collection of object classes which each model real world or abstract items by combining data to represent the item's properties with functions to represent the item's functionality. More specifically, an object is an instance of a programmer-defined type referred to as a class, which exhibits the characteristics of data encapsulation, polymorphism and

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

> inheritance. Data encapsulation refers to the combining of data (also referred to as properties of an object) with methods that operate on the data (also referred to as member functions of an object) into a unitary software component (i.e., the object), such that the object hides its internal composition, structure and operation and exposes its functionality to client programs that utilize the object only through one or more interfaces. An interface of the object is a group of semantically related member functions of the object. In other words, the client programs do not access the object's data directly, but must instead call functions on the object's interfaces to operate on the data.

A general description of "object-orientated programming principles" as in this portion of Hinson fails to suggest anything concerning a plurality of events or specific event within that plurality. There is no basis for taking this prior art description in Hinson of object-orientated programming principles and modifying the invention of Hinson. Accordingly, the reliance on this section of Hinson is further evidence that the obviousness rejection of Claim 12 is not well founded. The above discussion with respect to the combination of references with respect to Claim 1 is incorporated herein by reference. Applicants request reconsideration and withdrawal of the obviousness rejection of Claim 12.

With respect to the obviousness rejections of independent Claims 13, 24, 37, 38, 41, and 42, each claim includes at least the limitation as discussed above with respect to Claim 12. Therefore, the above remarks with respect to Claim 12 are applicable for each of these claims and are incorporated herein by reference. Applicants respectfully request reconsideration and withdrawal of the obviousness rejection of each of Claims 13, 24, 37, 38, 41, and 42.

Claims 15 to 19 have been cancelled and so render the obviousness rejections moot.

Claims 27 to 31 and 33 to 36 depended from Claim 26 and so distinguish over the prior art for at least the same reasons as

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

Claim 26. Applicants respectfully request reconsideration and withdrawal of the obviousness rejection of each of Claims 27 to 31 and 33 to 36.

Claims 1 to 6, 8 to 13, 24, 26 to 31, 33 to 38, and 40 to 42 remain in the application. Claims 1, 12, 13, 24, 26, 37, 38, and 40 are amended. Claims 15 to 19 are cancelled. Claims 7, 14, 20 to 23, 25, 32, 39, and 43 were previously canceled. For the foregoing reasons, Applicants respectfully request allowance of all pending claims. If the Examiner has any questions relating to the above, the Examiner is respectfully requested to telephone the undersigned Attorney for Applicant(s).

Respectfully submitted,

CERTIFICATE OF TRANSMISSION
I hereby certify that this correspondence is being facsimile transmitted to the U.S. Patent and Trademark Office, Fax No. 571-273-8300, on August 31, 2006.

Mona Marshall
Mona Marshall

August 31, 2006
Date of Signature

Forrest Gunnison
Attorney for Applicant(s)
Reg. No. 32,899

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

NIST
National Institute of Standards and Technology

Attachment 1
1/2

abstract data type

(definition)

Definition: A set of data values and associated operations that are precisely specified independent of any particular implementation.

Also known as ADT.

Specialization (... is a kind of me.)
dictionary, stack, queue, priority queue, set, bag.

See also *data structure.*

Note: Since the data values and operations are defined with mathematical precision, rather than as an implementation in a computer language, we may reason about effects of the operations, relations to other abstract data types, whether a program implements the data type, etc.

One of the simplest abstract data types is the stack. The operations new(), push(v, S), top(S), and popOff(S) may be defined with axiomatic semantics as following.

1. *new() returns a stack*
2. *popOff(push(v, S)) = S*
3. *top(push(v, S)) = v*

where S is a stack and v is a value. (The usual pop operation is a combination of top, to return the top value, and popOff, to remove the top value.) Contrast this with the axiomatic semantics definition of a set, a dictionary, or a queue.

From these axioms, one may define equality between stacks, define a pop function which returns the top value in a non-empty stack, etc. For instance, the predicate isEmpty(S) may be added and defined with the following additional axioms.

4. *isEmpty(new()) = true*
5. *isEmpty(push(v, S)) = false*

After Nell Dale <ndale@cs.utexas.edu> May 2001.

Author: PEB

Go to the Dictionary of Algorithms and Data Structures home page.

If you have suggestions, corrections, or comments, please get in touch with Paul E. Black.

Entry modified 10 February 2005.
HTML page formatted Tue May 30 08:36:55 2006.

Cite this as:
Paul E. Black, "abstract data type", in *Dictionary of Algorithms and Data Structures* [online], Paul E. Black, ed., U.S. National Institute of Standards and Technology. 10 February 2005. (accessed TODAY) Available from: http://www.nist.gov/dads/HTML/abstractDataType.html

NIST HOME

Attachment I
2/2